Introduction to SQL
Class 2

SELECT statement

- Bread and butter if you are working on databases
- Syntax
  - **SELECT** Column_Name_1, Column_Name_2, ..., Column_Name_N **FROM** Tab le_Name;
- What if I want to select all the columns
  - SELECT * FROM table_name;

But I don’t want to select everything, it doesn’t make any sense to me.

Data is garbage, information is gold

"Why did the data cross the road?
To get to the other side and become information!

SELECT Statement with WHERE clause

- Returns only the rows that satisfy certain conditions
- Syntax
  - SELECT * FROM Name_of_Table WHERE [condition];
- Conditions are written using SQL logical or comparison operator
- Examples

SELECT Statement so far

- SELECT ….
- FROM …..
- WHERE ….

SELECT Statement with ORDER BY clause

- The ORDER BY clause is used in SQL to sort the result set of a SELECT statement in ascending or descending order
- Syntax

```
SELECT column1, column2, ...
FROM table
ORDER BY column1 [ASC | DESC], column2 [ASC | DESC], ...;
```

- By default, the sort order is ascending (ASC)
- To sort the result set in descending order, we can use the DESC keyword.
- We can also use the numbers in the ORDER BY clause to indicate the column number instead of the column name.
    - Example

```
SELECT first_name, last_name
FROM customers
ORDER BY 2, 1 DESC;
```

SELECT Statement so far

- SELECT ….
- FROM …..
- WHERE ….
- ORDER BY

SQL SELECT Statement with GROUP BY clause

- Is used in SQL to group rows that have the same values in one or more columns.
- The result is a set of summary rows by the values in one or more columns.
- Syntax

```
SELECT column1, column2, aggregate_function(column3)
FROM table1
GROUP BY column1, column2;
```

- Aggregate functions are functions that are used to calculate some value
  - Example – COUNT,SUM,AVG,MIN,MAX
- Column name that is not included in the GROUP BY clause must be used in an aggregate function.
- Can also be combined and used with a where clause

SELECT Statement so far

- SELECT ….
- FROM …..
- WHERE ….
- GROUP BY …
- ORDER BY …

SELECT Statement with GROUP BY and HAVING clause

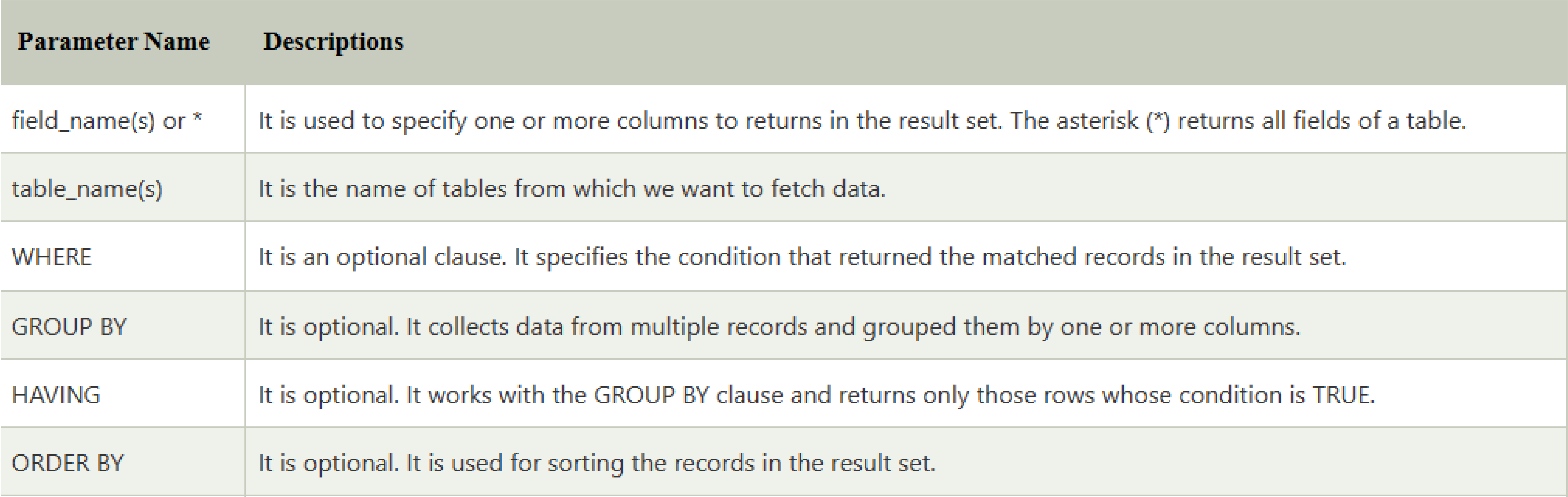
- The HAVING clause is used in SQL in conjunction with the GROUP BY clause to filter groups of rows based on aggregate values
- It is similar to the WHERE clause, but it is used to filter groups of rows, rather than individual rows.
- Syntax

```
SELECT column1, aggregate_function(column2)
FROM table
GROUP BY column1
HAVING aggregate_function(column2) operator value;
```

- It is important to note that the HAVING clause must come after the GROUP BY clause and before the ORDER BY clause if used
- We can use any column in the SELECT list that is included in the GROUP BY clause or an aggregate function.

SELECT Statement so far

- SELECT ….
- FROM …..
- WHERE ….
- GROUP BY …
- HAVING…
- ORDER BY …



Parameter Name	Descriptions
field_name(s) or *	It is used to specify one or more columns to returns in the result set. The asterisk (*) returns all fields of a table.
table_name(s)	It is the name of tables from which we want to fetch data.
WHERE	It is an optional clause. It specifies the condition that returned the matched records in the result set.
GROUP BY	It is optional. It collects data from multiple records and grouped them by one or more columns.
HAVING	It is optional. It works with the GROUP BY clause and returns only those rows whose condition is TRUE.
ORDER BY	It is optional. It is used for sorting the records in the result set.

SQL Operators (Can be used in MySQL as well)

- Reserved words and characters that can be used with WHERE clause in SQL Query
- There are two types of operators
  - Unary Operator
    - Syntax
      - Operator SQL_Operand
  - Binary Operator
    - Syntax
      - Operand1 SQL_Operator Operand2

SQL operators are categorized in the following categories:

1. SQL Arithmetic Operators
2. SQL Comparison Operators
3. SQL Logical Operators
4. SQL Set Operators
5. SQL Bit-wise Operators
6. SQL Unary Operators

SQL Arithmetic Operators

1. SQL Addition Operator (+)
2. SQL Subtraction Operator (-)
3. SQL Multiplication Operator (*)
4. SQL Division Operator (/)
5. SQL Modulus Operator (%)

Connect with me on LinkedIn

Please subscribe to our YouTube channel

Check out my GitHub profile

Follow me on Twitter(X)